Exploração de Computação Híbrida com OpenACC em um Algoritmo Friends-of-Friends para Classificação de Objetos Astronômicos

Ana Luísa V. Solórzano[1], **Andrea S. Charão**[1]
Renata S. R. Ruiz[2], **Haroldo F. de Campos Velho**[2]

[1] Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria

[2]Laboratório Associado de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais

Resumo. *A utilização de dispositivos aceleradores para processar porções de um código mostra-se promissora na área de computação híbrida. Neste trabalho, utilizou-se a ferramenta de paralelização com aceleradores OpenACC em um sistema composto por CPU e GPU para o algoritmo Friends-of-Friends, que lida com grandes quantidades de dados. Os resultados indicam que o algoritmo pode ser explorado neste ambiente, porém com algumas limitações.*

1. Introdução

A utilização de computação híbrida vem demonstrando ser uma opção vantajosa para resolver problemas que lidam com amplas quantidades de dados. Esse fato pode ser bem explorado para códigos em que certas porções se comportam melhor em CPU, enquanto outras têm maior potencial se executarem em dispositivos aceleradores.

Atualmente, uma plataforma de aceleração bastante utilizada é a GPU, mesmo existindo outras como FPGAs e Intel® Xeon Phi®. Dentre as diversas ferramentas para paralelização de programas com aceleradores estão CUDA, OpenCL e OpenACC. Neste trabalho, optou-se pelo padrão OpenACC, baseado no uso de diretivas para distribuir a computação em um ambiente híbrido.

Um cenário que lida com problemas de grande envergadura é a análise computacional de dados astronômicos provenientes de simulações cosmológicas e de observatórios virtuais. Considerando sua importância para, por exemplo, prever comportamentos similares na formação de grandes estruturas observadas no Universo, como galáxias e aglomerados de galáxias [Ruiz 2011], o desenvolvimento de programas eficientes e com um baixo tempo de execução é primordial.

Um dos algoritmos mais utilizados em simulações de N-corpos para classificar galáxias e aglomerados de galáxias é o Friends-of-Friends (FoF) [Huchra e Geller 1982, Caretta et al. 2008]. Em trabalhos anteriores, um algoritmo FoF sequencial foi implementado e paralelizado utilizando MPI para execução em um *cluster* [Ruiz et al. 2009], e posteriormente foi paralelizado com OpenMP [Berwian et al. 2017]. Entretanto, o seu desempenho em um ambiente de computação híbrida ainda não havia sido explorado.

2. Algoritmo Friends-of-Friends (FoF)

O algoritmo FoF é utilizado para identificar estruturas do Universo através do agrupamento de partículas com uma dada proximidade física [Huchra e Geller 1982, Caretta et al. 2008]. Para isso, ele considera que se dentro de uma esfera de raio de percolação R, ao redor de uma partícula, existirem outras partículas, elas são consideradas todas pertencentes ao mesmo grupo e chamadas de amigas. Após, ele realiza o mesmo procedimento para uma esfera ao redor de cada amiga, até que nenhuma nova amiga possa ser adicionada ao grupo.

A execução do algoritmo recebe como parâmetro um arquivo com um grande volume de dados, em que cada linha contém informações relativas a uma partícula e a primeira linha define o número de partículas a serem classificadas, e um valor para o raio de percolação. O tamanho do raio de percolação varia de acordo com os objetos astronômicos a serem identificados, utilizando-se 0.1 para identificar galáxias, 0.18 para aglomerados de galáxias e 0.92 para super aglomerados de galáxias [Caretta et al. 2008].

3. Paralelização do FoF com OpenACC

Optou-se por utilizar a ferramenta OpenACC, pois ela possibilita a geração de códigos paralelizados para placas gráficas aceleradoras sem a necessidade de maior conhecimento quanto a sua arquitetura, usando diretivas de compilação. Isso permite gerar um código com poucas modificações em comparação ao original.

A paralelização foi feita no laço mais interno do algoritmo (Algoritmo 1), visto que ele não possui dependência de dados. Para isso, foi utilizada a diretiva `#pragma acc parallel`, que descreve uma região do código a ser acelerada com os *kernels* da GPU. Notou-se que a variável `dist`, que armazena o valor da distância entre duas partículas, deve ser privada para evitar valores inconsistentes.

```
for (i = 0 ; i < N ; i++){
  k++;
  while (igru[i] != 0 )i++;
  igru[i] = k;
  for (j = i ; j < N ; j++){
    if(igru[j] == k) {
      for (l = (i + 1) ; l < N ; l++){
        if (igru[l] == 0){
          dist = sqrt((x[j] - x[l])*(x[j] - x[l]) +
          (y[j] - y[l])*(y[j] - y[l]) +
          (z[j] - z[l])*(z[j] - z[l]));
          if (dist <= rperc){
            igru[l] = k;
          }
        }
      }
    }
  }
}
```

Algoritmo 1. Versão sequencial

Visto que o laço paralelizado lê os vetores `x`, `y`, `z` e `igru`, e também pode alterar o último, utilizou-se a diretiva `#pragma acc data copyin` antes do laço mais externo, de modo a copiar uma única vez para a GPU os valores dos vetores apenas lidos. O `igru` precisou de maior atenção, pois é modificado antes de ser lido no laço paralelizado e pode ser modificado dentro dele. Esse tratamento gerou duas abordagens semelhantes:

1. A primeira consistiu em utilizar a diretiva `#pragma acc copy`, que copia o vetor da CPU para a GPU antes de entrar na região paralela e da GPU para a CPU antes de sair, garantindo a consistência dos dados do vetor durante toda a execução.
2. A segunda utilizou a diretiva `#pragma acc update device`, que copia os valores atualizados no `igru` da memória local para o dispositivo após sua alteração no laço mais externo, e `#pragma acc update self`, que atualiza os dados modificados no dispositivo para a memória local após sair da região paralela. Para isso, precisou-se alocar o vetor no acelerador usando a diretiva `#pragma acc data create` antes do laço mais externo.

4. Experimentos e Resultados

Os experimentos foram realizados em um servidor com processador Intel® Xeon® E5620 de quatro cores físicos e oito virtuais, com 32KB de cache L1, 256KB de cache L2 e 12MB de cache L3, e 12 GB de memória, e com uma GPU NVIDIA GeForce GTZ Titan X. O sistema operacional é Debian 9, versão do Linux 4.9.0-2, e o compilador utilizado foi o da Portland Group (PGI) edição Community [NVIDIA] versão 17.4-0.

A compilação do código original apresentou alguns erros, acusando sobrecarga de funções. Possivelmente, isso ocorreu porque a `libc` nativa define as funções das bibliotecas declaradas no mesmo momento em que a `libstdc++` também as carrega, assumindo que a libc não o faça. Assim, a solução encontrada foi não utilizar as bibliotecas conflitantes `math` e `stdlib`, adaptando um trecho do código que utilizava a função `sqrt()` para cálculo da raiz quadrada.

A partir de três arquivos de entrada com 65536 partículas (Arquivo 1), 174761 partículas (Arquivo 2) e 1048576 (Arquivo 3), foram realizadas 30 execuções dos arquivos 1 e 2 e 10 execuções do Arquivo 3, todas utilizando raio de percolação 0.1. Os resultados obtidos são apresentados na Tabela 1.

Arquivo	Média (s) serial	Média (s) versão 1	Speedup versão 1	Média (s) versão 2	Speedup versão 2
1	25.25	21.07	1.20	19	1.33
2	162	126.31	1.28	112.40	1.44
3	5120.30	4015.29	1.27	3582.66	1.43

Tabela 1. Análise de desempenho para as duas abordagens com OpenACC

Utilizando o *profiler* nvprof, investigou-se a diferença entre os *speedups* das duas abordagens. Na segunda, o *pragma* de atualização de dados da CPU para a GPU apenas sincroniza os valores alterados, enquanto que na primeira, o *pragma* de cópia escreve todo o vetor na GPU a cada iteração do laço, resultando em um maior processamento. Com essa análise, também se constatou que o maior tempo gasto em ambas abordagens,

cerca de 60% do tempo total, está no bloqueio da CPU a espera dos *kernels* lançados pelo acelerador, ou seja, tempo em que a GPU está fazendo o seu trabalho.

Os resultados de desempenho demonstraram que, apesar de fazer uso do processamento paralelo em GPU, o programa ficou distante de explorar todo o potencial oferecido pela placa. Esse fato, sustenta-se na lógica utilizada no algoritmo FoF, que contém várias estruturas condicionais e dependências de dados, o que dificultou a sua paralelização no dispositivo utilizado.

Também, o laço mais interno possui um desbalanceamento de carga, o que levou à utilização de diretivas de escalonamento para determinar a quantidade de dados a serem processados por *kernel* lançado. Entretanto, não obteve-se êxito, visto que a computação no laço paralelizado não depende apenas do número de partículas a serem processadas, mas também das informações quanto as suas posições, o que implicaria em cargas distintas para cada arquivo de entrada.

5. Considerações Finais

A partir dos resultados, observou-se que o algoritmo Friends-of-Friends pode explorar o uso de uma GPU como acelerador, porém não faz uso de todo o seu potencial. Por outro lado, embora OpenACC não proporcione ao desenvolvedor acesso a detalhes sobre o dispositivo, ela é uma ferramenta simples de ser utilizada e de alta portabilidade, sendo vantajosa para investigações iniciais. Assim, para trabalhos futuros, poderiam ser investigadas outras alternativas envolvendo a reescrita do algoritmo FoF, visando o seu processamento paralelo em um sistema de computação híbrida composto por CPU e GPU.

Referências

Berwian, L., Zancanaro, E. T., Cardoso, D. J., Charão, A. S., da Rocha Ruiz, R. S., e de Campos Velho, H. F. (2017). Comparação de estratégias de paralelização de um algoritmo friends-of-friends com openmp. In *Anais da XVII Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*, pages 279 – 252.

Caretta, C. A., Rosa, R. R., de Campos Velho, H. F., Ramos, F. M., e Makler, M. (2008). Evidence of turbulence-like universality in the formation of galaxy-sized dark matter haloes. *Astronomy & Astrophysics*, 487(2):445–451.

Huchra, J. P. e Geller, M. J. (1982). Groups of galaxies. I - Nearby groups. *Astrophysical Journal*, 257:423–437.

NVIDIA. PGI community edition. Available: https://developer.nvidia.com/openacc-toolkit.

Ruiz, R. S. d. R. (2011). *Turbulência em cosmologia: análise de dados simulados e observacionais usando computação de alto desempenho*. PhD thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

Ruiz, R. S. R., Campos Velho, H. F., e Caretta, C. A. (2009). Parallel algorithm friends-of-friends to identify galaxies and cluster of galaxies for dark matter halos. In *Proceedings...* Workshop dos Cursos de Computação Aplicada do INPE, 9. (WORCAP)., Instituto Nacional de Pesquisas Espaciais (INPE).